REMARKS

Applicants respectfully request reconsideration and allowance of all pending claims in view of the above-amendments and the following remarks.

I. CLAIM REJECTIONS UNDER §102(e)

Claims 1, 2, 10, 13 and 14 were rejected under §102(e) as being allegedly anticipated by Pajak et al. U.S. Publication No. 2003/0145286A1.

A. **Present Application**

In an example of the present application, the intent is to find design files that are located somewhere in a directory structure within a first environment (e.g., a hardware description environment), create a list of those files, and feed them to a tool in a second environment. In one example, and contrary to Pajak, the need for hard-coded paths in shell script usage is removed. For example, RTL files could be located in several sub-directories in the directory structure and a designer may want to synthesize them to create a gate-level netlist. The .rmk files are used to define which files are the RTL files. In general, the tool finds these .rmk files, figures out which design files are the RTL files (or whichever design files have been requested to be found/gathered), and returns a list of the RTL files including the full path name to the design file in the existing directory structure within the first environment so that list can be fed to and used by the synthesis tool in the second environment for accessing the design files.

Referring to claim 1, the directory structure is parsed to locate file paths to the description files. Also, the description files are parsed to identify file paths to the design files that are defined by the description file. An index is generated, which correlates each description file and its respective file path in the first environment.

A list of the design files is constructed, which contains design file names and respective full file paths for each of the design files in the first environment. The respective full file paths are constructed by concatenating the file path of the description file that is identified in an index to the file path of the design file that is defined by the description file.

Thus, in the example discussed above, the synthesis tool (in the second environment) accesses at least one of the design files within the first environment through the respective full

file path to the design file in the first environment. This full file path was produced by concatenating the file path of the description file identified in an index to the file path of the design file defined by the description file.

B. **Pajak**

Pajak discloses a self-contained embedded test design environment and environment setup utility, which is difference from the inventions recited in the claims of the present application. In Pajak, the user takes a user created file and runs the environment setup utility to get a template for this file, called a workspace configuration file. The designer then edits the configuration file. Figures 9-11 show examples of the configuration file.

When the user runs the setup utility, it creates directories (Figure 8-repository generator), creates links (Figure 8-soft link generator), and creates the makefiles (Figure 8-process control file generator), etc.

The user runs the makefiles (process control files) to perform the function desired (test insertion, verification, etc.). Or the user can do a 'make all', that runs everything. This is the same as the user running the individual makefiles in the proper order.

C. **Differences Between Pajak and An Example Embodiment of the Present Application**

Without focussing on the claims for the moment, the Pajak system is much different than that disclosed in the present application, according to at least one embodiment (referred to as "the present disclosure"). The following section should be used as an example for background information and not to limit any particular claim.

These differences include:

1. The present disclosure parses the directory structure of a system, finding the .rmk files (description files) to define the path from the root directory for files. Pajak has the user define these paths in the workspace configuration file (see Figure 10). In the present disclosure, the description files specify the files and possibly the path to the file relative to the description file location, but not the full path from the root directory of the system.

2. The present disclosure creates an index file that correlates the full paths and files.

3. Contrary to Pajak, the present disclosure does not create a directory structure (the repository) and set up links. The present disclosure does not need the links because it has the index file, etc.

4. Pajak says their system is portable because the repository can be moved using tar -h. The command, "tar" represents "create tape archive". The tar -h option will (as described from the UNIX manual "man" page): "Follow symbolic links as if they were normal files or directories. Normally, tar does not follow symbolic links."

So the user gets all the files in the tarbar instead of links that will not resolve when the user "untars" it somewhere else. In contrast, the present disclosure is portable wherein the user can move it, remove the index file, and let the index file be regenerated to find everything that is now in the new location.

5. Pajak suggests their system can use relative paths in their make files (processor control files). This works because Pajak sets up the links in the setup utility . . . because a user had to define the hard/complete paths in the workspace configuration file. As described above, the user of the present disclosure does not need to define the hard/complete paths. The location of the description files will define the paths.

D. **Examiner's Comments Regarding Claim 1**

The Examiner cited various paragraphs and figures in an effort to support the rejection of claim 1. These citations are discussed below.

Regarding claim 1, step a), Pajak [0035] is just saying that they have a variety of files that need to be stored and a directory structure needs defined to store these files (Figure 4).

Regarding step b), claim 1 refers to parsing the directory structure to find the description files, whereas Pajak [0091] refers to parsing the input workspace configuration file and creating the directories, links & makefiles. So different things are going on. The only commonality is the use of the term "parse".

Regarding step c), claim 1 refers to creating the index file. Pajak [0109] refers to running the top-level makefile (processor control file) with a target of "All" to cause all the lower-level makefiles to be run in the correct order. Applicants do not understand the relationship the

Examiner is trying to make with respect to claim 1 of the present application.

Regarding step d), claim 1 refers to constructing a list of the paths & design file names. Pajak [0122] refers to what is in the input file and how the generators use this - to create corresponding directories.

Also regarding step d), claim 1 defines the full file paths for each of the design files in the list as being "are constructed by concatenating the file path of the description file that is identified in the index to the file path of the design file that is defined by the description file." As described above, in the present disclosure, the description files specify the design files and the path to the design files relative to the description file location, but not the full path from the root directory of the system.

Pajak [122] simply refers to "provides . . . the location of its concatenated netlist . . .." Thus, it is the netlist that is concatenated. This paragraph has nothing to do with concatentating file paths, much less in the manner recited in claim 1. Specifically, Pajak does not disclose concatenating the file path of the description file that is identified in the index to the file path of the design file that is defined by the description file, as recited in claim 1.

Regarding step e), claim 1 refers to accessing the design file by a design tool in a second environment. The Office Action cites paragraphs [0081]. [0082], [0122] and Figure 10. Applicants believe the Examiner is trying to pull paragraphs from Pajak that show the user input file being used to make the directories and links. The difference is that the present disclosure use the design file of interest, while Pajak is just setting up the directories, links and makefiles with relative paths so that the user can then run the makefiles to do something useful. With the present disclosure, for example, the user does not need to create the links and does not create directories to be able to access the design files. Rather, the present disclosure parses directories and makes an index file, not links, for example.

E. **Claims Not Anticipated By Pajak**

For at least the foregoing reasons, Pajak does not anticipate each and every element of claim 1, or of claims 2, 10, 113 and 14 for similar reasons.

Applicant respectfully requests that the rejection of these claims under §102(e) based on

Pajak be withdrawn.

The Director is authorized to charge any fee deficiency required by this paper or credit any overpayment to Deposit Account No. 12-2252.

Respectfully submitted,

WESTMAN, CHAMPLIN & KELLY, P.A.

By: /David D. Brush/

David D. Brush, Reg. No. 34,557
900 Second Avenue South, Suite 1400
Minneapolis, Minnesota 55402-3319
Phone: (612) 334-3222 Fax: (612) 334-3312